Javascript

- Javascript which is often known as JS, is a high-level dynamic interpreted programming language.
- It allows client-side scripting to create completely dynamic web applications & websites.
- It can be executed on the browser as well as the server
- Javascript is a safe language when used in the browser
- These are languages that get "transpiled" to Javascript

* Javascript was invented by Brendan Eich in 1995.
* JS code is inserted between <script> & </script> tag
* scripts can be placed in the <body>, or in the <head> section of an HTML, page or in both
* we can place the external script reference in <head> or <body> as we like
* JS displays possibilities
  - using innerHTML
    JS can use the document.getElementById(id) method
  - using document.write()
    document.write() methods should be used only for testing
  - using window.alert()
    we can skip the window keyword
  - using console.log()
    for debugging purpose, we can use console.log() in the browser to display data

- innerHTML
  - document.getElementById(id)

- document.write()
- window.alert() OR alert()
- console.log()

* we can use window.print() method in the browser to print the content of current window.

* Ending statements with semicolons is not required but highly recommended.

Syntax

* JS values
  - fixed values or Literals
  - Variable values or Variables
    ↓
    used to store data

* All JS identifiers are case-sensitive
* JS uses unicode character set.

Comments

Single line comment - //
Multiline comment - /* */

Variables

* 3 ways to declare JS variable
  - using var - declare a variable
  - using let - declare a block variable
  - using const - declare a block constant

* Let

- Variables defined with let cannot be Redeclared.
- Variables defined with let have block scope

  Ex- let x = "ash"
      let x = 0

  Variable declared inside block scope {} block cannot be accessed from outside the block

  {
  let = 2 cannot be accessed
  }

  { var = 2 can be accessed from outside
  }

- let hoisting
  - variables defined with var are hoisted to the top and can be initialized at any time
  - using let variable before it is declared will result in a referenceerror

* const

- variables defined with const cannot be redeclared & reassigned
- variables defined with const have block scope.
- Js const variables must be assigned a value when they are declared.

  declaring a variable with const is similar to let when it comes to block scope.

- const Hoisting

  using a const variable before it is declared will result in referenceerror

* Data types

- let x = 16 + 4 + "Ash";
  result - 20Ash
- let x = "Ash" + 16 + 4;
  result - Ash164
- when addin a number & a string, Js will treat the number as a string

In the 1st example, Js treats 16 & 4 as numbers until it reaches "Ash".

In the 2nd example, since the first operand is a string, all operands are treated as string

* Js types are Dynamic

this means that the same variable can be use to hold different data types.

```
let x;
x = 5;
x = "Ash";
```

* JS Booleans

- booleans can only have two values: true & false
- they are often used in conditional testing

* The typeof operator

We can use the JS typeof operator to find the type of Js variable

* Undefined

- In Js, a variable without a value, has the value undefined. The type is also undefined
- let car = undefined;
  let car; > result - undefined

* Empty string
  - An empty string has both a legal value & a type
  - let car = "";
    result - value =
    type of = string

* JS objects
  - objects are variables too. but objects can contain many values
  - It is a common practice to declare objects with the const keyword.
  - ex -

```
var names = {
    ravi: 34,
    sub: 78,
    harry: 99.97
};
```

  - objects methods
  - methods are action that can be performed on objects
  - methods are stored in properties as function definations.
  - this keyword
    - this refers to the owner of the functions
    - this is the person object that "owns" the fullname function
    - this.firstname means the firstname property of this object
    - const person = {

```
firstname : "ravi",
lastname : "pal",
id = "56",
fullname : function() {
```

return this.firstname + " " + this.lastname;
}
};

result → Person.fullname() - Ravi Doe

* A very high level - 2 types of data types in JS

1) Primitive datatypes
   - undefined
   - null
   - number
   - string
   - boolean
   - symbol

2) Refrence data types
   - Arrays
   - Objects

Arrays

* an array is a special variable, which can hold more than one variable

```
const array_name = [item1, item2 ...];
```

* Arrays are objects
  - arrays are special type of objects. The type of operator in JS returns "object" for arrays.
  - const Person = ["Ash", "Doe", 46]
  - const Person = {firstname: "Ash", lastname: "Doe", age: 46};

* properties
  - cars.length - returns the no. of length
  - cars.sort() - sort the array
  - cars[cars.length-1]; - access last element
  - Array.forEach() - looping
  - cars.push(" ") - add elements
  - cars[cars.length] = "Audi"; adds audi > same

* Difference in assays & objects

assays use numbered indexes	objects use named indexes
use assay when we want the element name to be string	use objects when we want the element names to be numbers.

* const points = new Array(); - Bad
  const points = []; - good

* How to recognize an array
  - Array.isArray(fruits); → True
  - fruits instanceof Array; → True

* Methods
  - converting arrays to strings
    - document.getElementById("demo").innerHTML = fruits.toString();
    - document.getElementById("demo").innerHTML = fruits.join("*");
  - shifting is equivalent to popping working on first element
    - fruits.shift()
  - unshift() methods add new element at first
    - fruits.unshift("11");

- delete.fruits[0]; - changes the 1st element in fruits to undefined.
- splice() method can be used to add new elements to the array.
  - fruits.splice(2, 0, "lemon", "kiwi");
    - 2 → At what position
    - 0 → how many items should be deleted
  - fruits.splice(0, 1); - removes the first element
- concat() methods create a new array by merging existing array.
  - const my = mygirls.concat(myboys);
  - const a = b.concat(c,d);
- the slice() methods slices out piece of an array into a new array.
  - the slice() method creates a new array. It doesn't ~~change~~ ~~or~~ remove any elements from the source array.
  - slice(1, 3)
    - 1 → from
    - 3 → upto
- fruits.sort()
- fruits.reverse()
- points.sort(function(a,b){return a-b});
- points.sort(function(a,b){return b-a});
- math.max.apply(null, arr); deciding order

Functions

* A JS function is executed when "something" invokes it.

```
function myFunction(p1, p2) {
    return p1 * p2
}

c1 = myFunction(4, 6)
```

Conditional statements

* ```
  if (condition)
    { }
  else
    { }
  ```

* ```
  if (condition) { }
  else if (condition 2) { }
  else { }
  ```

* Switch statement

```
switch(condition) {
case x:
    // code block
    break;
case y:
    // code block
    break
default:
    // code block
}
```

Loops

* for - loops through a block of code a number of times

```
for (statement 1; statement 2; statement 3)
  {
    // code block
  }
```

* The for in loop

the JS for in statement loops through the properties of an object.

```
for (key in object) {
    // code
}
```

* The for of loop

the JS for of statements loops through the values of an iterable objects

```
for (variable of iterable) {
    //code
}
```

* While loop

```
while (condition)
  {
    //code
  }
```

* do while loop

```
do { }
while (condition);
```

* arr.forEach(function(element) {
  console.log(element)
})

Document Object Model (DOM)

- The DOM is a W3C (World Wide Web Consortium) Standard
- The W3C DOM is a platform & language-neutral interface that allows progsams & scripts to dynamically access & update the content, structure & style of a document.

* HTML DOM
  - the HTML elements as objects
  - the properties of all HTML elements
  - the methods to access all HTML elements
  - the events for all HTML elements

* document.getElementById("demo").innerHTML = "Hi";
  - getElementById is a method
  - innerHTML is a property
  - the innerHTML property can be used to get or change any HTML element, including <html> & <body>.

* Finding HTML elements
  - document.getElementById(id)
  - document.getElementsByTagName(name)
  - document.getElementsByClassName(name)

* Change HTML elements

element.innerHTML = new html content
element.attribute = new value
element.style.property = new style

* Adding & Deleting elements
  - document.createElement(element)
  - document.removeChild(element)
  - document.appendChild(element)
  - document.replaceChild(new, old)
  - document.write(text)

* Adding events handlers
  - document.getElementById(id).onclick = function()
    {code}

* DOM elements
  - Finding HTML elements by id
    - const element = document.getElementsByTagName("intro")
  - Finding HTML elements by Tag name
    - const element = document.getElementsByTagName("p");
  - Finding HTML elements by class name
    - const x = document.getElementsByClassName("intro");
  - Finding HTML elements by CSS selector
    - const x = document.querySelectorAll("p.intro");
  - Finding HTML elements by HTML object collections.

Set time out & set interval

* arrow function

function sum(a,b){
    return a+b;
}

sum = (a,b) => {
    return a+b;
}

* setTimeout (logKaro ; 2000);
                    ↑ milliseconds

```
logKaro := () => {
    console.log("I m ur log")
}
```

↓ It will print after 2 sec

* setInterval (logKaro, 2000);

→ copy like aise

↓

after every 2 sec output will print

Errors

* try - test a block of code for errors
  catch - lets you handle the error
  throw - lets you create custom errors
  finally - lets you execute code, after try & catch

* try & catch

```
try {
    block of code to try
}
catch(err) {
    block of code to handle error
}
```

* Throw
- the throw statement allows you to create a custom error
- Technically you can throw an exception (throw an error)
- throw "too big"; - throw text
  throw 500; - throw number

* finally
- the finally statement lets you execute code after try & catch
- try {
    block of code to try
  }
  catch(err) {
    block of code to handle errors
  }
  finally {
    block of code to be executed regardless of the try/catch result
  }

* Error names
- EvalError - error occured in eval function
- RangeError
- ReferenceError
- SyntaxError
- TypeError
- URIError - an error in encodeURI() has occurred

JS JSON (Javascript object Notation)

- Json is a for storing & transporting data
- Json is often use when data is sent from a server to a web page
- Json is a lightweight data interchange format
- Json is language independent *
- Json is self describing & easy to understan.

* Rules

- Data is in name/value pairs
- Data is separated by commas
- curly braces hold objects
- squre brackets hold assays
- always in double quotes

* example

```
{
    "employees" : [
    {"firstname" : " John", "lastname" : "Doe"},
    {"firstname" : "Anna", "lastname" : "smith"},
    {"firstname" : "Peter", "lastname" : "Jones"}
]
}
```

Vessions

ECMAScript is the official name of the lang. that is used to maintain the standard of JS